Energy-Efficient Service Function Chain Provisioning

Nicolas Huin, Andrea Tomassilli, Frédéric Giroire, Brigitte Jaumard

Energy-Efficient Service Function Chain Provisioning

Nicolas Huin, Andrea Tomassilli, Frédéric Giroire, Brigitte Jaumard

***Abstract*—Network Function Virtualization (NFV) is a promising network architecture concept to reduce operational costs. In legacy networks, network functions, such as firewall or TCP optimization, are performed by specific hardware. In networks enabling NFV coupled with the Software Defined Network (SDN) paradigm, Virtual Network Functions (VNFs) can be implemented dynamically on generic hardware. This is of primary interest to implement energy efficient solutions, in order to adapt the resource usage dynamically to the demand. In this paper, we *study how to use NFV coupled with SDN to improve the energy efficiency of networks.* We consider a setting in which a flow has to go through a Service Function Chain, that is several network functions in a specific order. We propose an ILP formulation, an ILP-based heuristic, as well as a decomposition model that relies on joint routing and placement configuration to solve the problem. We show that virtualization provides between 22% to 62% of energy savings for networks of different sizes.**

I. Introduction

With the large yearly increase of Internet traffic and the growing concern of the public and governments towards greenhouse gas emissions, future networks will have to be more energy efficient [1]. In the past few years, this has been the focus of extensive research work [2]. One of the classic methods to reduce the energy consumption of networks is to try to aggregate network traffic on a small number of network equipment in order to put to sleep the unused hardware. However, an additional challenge is given by the fact that today's traffic must pass through a certain number of network functions. Examples of network functions include deep packet inspection (DPI), firewall, load balancing, and WAN optimization. The network functions often need to be applied in a specific order, e.g., in a security scenario, the firewall has to be applied before carrying out a DPI, as the latter is more CPU intensive than the former. In this context, a *Service Function Chain (SFC)* is a list of network functions, that need to be applied to a flow in a particular order. These functions are carried out by specific hardware, which are installed at specific locations of the network. The paths followed by demands are thus very constrained, reducing the opportunities to aggregate traffic.

With the emergence of techniques of Network Function Virtualization (NFV), the functions can now be executed by generic hardware instead of dedicated equipment. Coupled with the Software Defined Network (SDN) paradigm, NFV brings a great flexibility to manage network flows. Indeed, with the centralized control allowed by SDN, the flow can be managed dynamically from end-to-end and the service functions can be installed only along paths for which and when they are necessary. These new paradigms thus bear the opportunity for energy savings in networks.

In this work, we explore the potential energy savings of using NFV for Service Function Chains. We consider the problem of reducing network energy consumption while placing service functions using generic hardware along the paths followed by flows. A difficulty is that the network functions have to be executed in a specific order and can be repeated several time in the same chain.

In summary, the contributions of this work are the following

- We show how virtualization can be used to improve the energy efficiency of networks, when demands have to go through a chain of services. To the best of our knowledge, we are the first to propose such a method.
- We propose a way of modeling this problem based on Integer Linear Programming (ILP). The ILP can solve optimally instances of small sizes.
- To handle instances of larger sizes, we thus propose and validate a *heuristic algorithm*, GreenChains,
- and we formulate a Column Generation model.
- We provide enhancements of the model with the use of cuts, as the our problem is a difficult optimization problem. As a matter of fact, it contains a sharp On-Off phenomena, as a network device consumes a large portion of its energy as soon as it is used, even if very lightly used. Cuts allow the reduction of the integrality gap.
- This allows us to carry out extensive simulations on networks of different sizes. We study three different scenarios: a *legacy scenario* which serves as baseline for comparison, a *hardware scenario* in which the routing can be changed dynamically by a centralized SDN controller, but in which network functions are executed by specific hardware, and finally, an *NFV scenario* in which the network functions are virtualized and can be placed dynamically. We show that from 22% to 62% of energy can be saved during the night while respecting the constraints of the service chains.
- Finally, we propose a latency analysis to evaluate the impact on delays of switching off some network elements to save energy.

The article is organized as follows. In Section II, we review the current works on energy efficiency and Service Function

N. Huin's, A. Tomassilli's and F. Giroire's affiliation is Université Côte d'Azur, Inria, CNRS, I3S, UNS, Sophia Antipolis, France. Emails: nicolas.huin@inria.fr, andrea.tomassilli@inria.fr, frederic.giroire@cnrs.fr

B. Jaumard is with the department of Computer Science and Software Engineering, Concordia University, Montreal (QC) Canada, Email: bjaumard@cse.concordia.ca

Chaining. The problem is presented in Section III along with the power model and the layered graph model used in our mathematical formulations. We present in Sections IV, V and VI the ILP formulation, GREENCHAINS and column generation scheme, respectively. We then compare the models and assess their quality in Section VII.

II. RELATED WORK

A. Service Chains

Several works study the problem of service function chain placement, but taking other metrics or other scenarios into account. Savi *et al.* [3] proposes a different ILP model to solve the problem. They study the impact of the positions of the network functions on the processing costs. Gupta *et al.* [4], [5] explores the joint placement and routing of traffic in order to minimize the network bandwidth consumption. In Martini *et al.* [6], a layered ILP model close to the one we propose in the paper is proposed, but with latency minimization as optimization task. [7] explores the problem of joint optimization of maximum link, CPU core and maximum delay in the network while placing the VNFs. Last, Riggio *et al.* [8] considers a cloud environment in which the load has to be load-balanced in order to minimize the computation and the communication overheads. However, these works do not consider the problem of minimizing network energy consumption with a dynamic traffic.

Energy-Aware Routing. Several works have proposed algorithms to obtain energy aware routing, see e.g., Chiaraviglio *et al.* [9]. However, these works are hard to be put in practice as operators of legacy networks are reluctant to change their network configurations.

B. SDN and Network Energy Efficiency

Recently, researchers have started to explore how the introduction of the SDN paradigm with a centralized control and a live report of metrology data may enable dynamic routing. In particular, it would allow the implementation of energy-aware routing algorithms, as discussed in Giroire *et al.* [10]. However, these papers did not consider the constraints of network functions. Some particular works considered some specific class of network functions, like compression [11], but not the general problem of ensuring that flows are treated by the network functions.

C. Network Virtualization and Network Energy Efficiency

Only two papers explore the potential of network virtualization for energy efficiency. In Bolla *et al.* [12], the authors present an extension of an open source software framework, the Distributed Router Open Platform (DROP), to enable a novel distributed paradigm for NFV. DROP includes sophisticated power management mechanisms, which are exposed by means of the Green Abstraction Layer. In [13], authors estimate the energy savings that could result from the three main NFV use casesVirtualized Evolved Packet Core, Virtualized Customer Premises Equipment and Virtualized Radio Access Network. However, both papers do not consider the constraints of service chains.

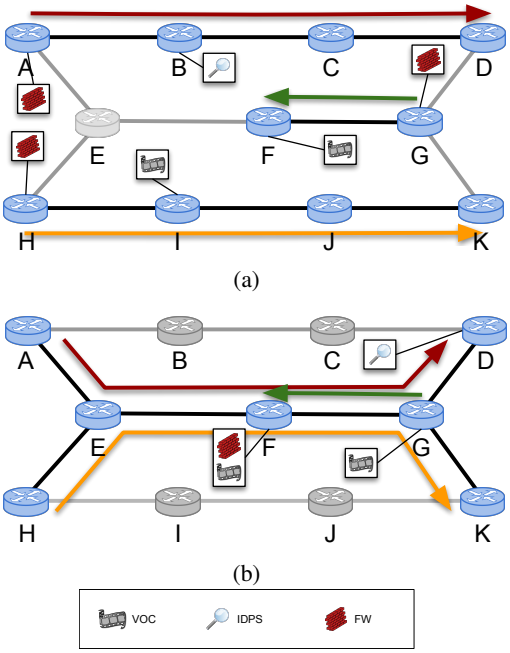


Fig. 1: Example of Energy efficient Service Function Placement. Greyed links and nodes are inactive.

III. STATEMENT OF THE PROBLEM: SFC AND VNF PLACEMENT

A. Notations.

We assume the network to be represented by a directed graph $G = (V, L)$, where V is the set of nodes (indexed by v), and L is the set of links (indexed by ℓ). Each node $u \in V$ has a set of computing, storage and network resources denoted by C_u to host network functions. Within this study, we assume that the resources are described by a given number of CPU cores.

Traffic is described by a set of requests D, in which each request d is defined by a 4-tuple (v_s, v_d, c, D^c_{sd}), where v_s is the source of the request, v_d its destination, D^c_{sd} its bandwidth requirement, and c the requested service chain. Indeed, each request d is associated with a given application, which is required to pass through a given SFC. Let F be the overall set of virtual functions arising in the service chains, indexed by f, and C be the set of service chains, indexed by c. Each service chain c corresponds to a sequence of n_c functions $f^c_1, \ldots, f^c_i, \ldots, f^c_{n_c}$, where f^c_i denotes the ith function of chain c. Note that some functions may appear more than once in a given chain. Each virtual function f has its one resource requirement, and we denote by Δ_f the number (fraction) of cores required by the function f per bandwidth unit.

The *Energy Efficient Service Function Chain Provisioning (*EE-SFCP*)* problem consists in jointly provisioning a set D of requests coupled with service function chains C and placing virtual functions arising in the chains, in order to minimize the network energy consumption, subject to link and node capacities.

Figure 1 shows examples of a Energy Efficient Service Function Chain provisioning problem. We have three requests

and two types of services. Demand $D_1 = (A \rightsquigarrow D)$ requires 1 unit of bandwidth and the execution of a firewall (FW) and a packet inspector (IDPS). Flows $D_2 = (H \rightsquigarrow K)$ and $D_3 = (G \rightsquigarrow F)$ need 1 unit of bandwidth, and the execution of a firewall followed by a video optimizer (VOC). An instance of a firewall uses $\Delta_{\text{FW}} = 0.33$ core per unit of bandwidth, an instance of a video optimizer requires $\Delta_{\text{VOC}} = 1$ core, and an instance of a packet inspector uses $\Delta_{\text{FW}} = 2$ cores. Each link has a capacity of 3 units of bandwidth, and each node hosts 2 cores.

In Figure 1a, flows are routed according to the shortest path between their source and destination. We need to place a firewall function instance on three different nodes (namely A, G, and H) to cover the three demands. Flows D_2 and D_3 get their video optimizer on nodes F and I, respectively. The packet inspector of flow D_1 is installed on node B. In this last configuration, five links can be shut down ((A, E), (H, E), (E, F), (G, D), (G, K)) and node E can be put to sleep. A total of 7 cores are active (1 on nodes A, F, G, H, and I, and two on node B).

However, there exists another routing that minimizes the energy consumed by the network. It allows the shutdown of one more link and the reduction of the number of active cores by two units. Indeed, if we consider the routing given in Figure 1b, we can group all the firewall instances on node F. Since nodes only host 2 cores, we need to put one video optimizer instance on node F, in charge of D_2; D_3 is served by the instance on node G. We now only use 5 cores in total, and we can now shutdown links (A, B), (B, C), (C, D), (H, I), (I, J), and (J, K).

B. Power Model

Campaigns of measures of power consumption (see, e.g., [14]) show that a network device consumes a large amount of its power as soon as it is switched on and that the energy consumption does not depend much on the load. Following this observation, on/off power models have been proposed and studied. Later, researchers and hardware constructors have proposed more energy proportional hardware models [15]. To encompass those different models, we use a hybrid power model in which the power of an active link ℓ is expressed as

$$P_\ell = P_\ell^{\text{ON}} + \frac{\text{BW}_\ell}{C_\ell^{\text{LINK}}} P_\ell^{\text{MAX}},$$

where P_ℓ^{ON} represents the energy used when the link ℓ is switched on, BW_ℓ the bandwidth that is carried on ℓ, and P_ℓ^{MAX} the additional energy consumed by ℓ when it is fully capacitated, i.e., when the amount of carried bandwidth equals the transport capacity (C_ℓ^{LINK}) of link ℓ.

We assume that links can be put into sleep mode, by putting to sleep both endpoint interfaces. Two links in opposite direction between a pair of nodes are assumed to be in the same state (active or in sleep mode), as the send and receive elements of a unidirectional fiber are usually controlled by the same interface. Routers cannot be put into sleep mode, as there are the sources/destinations of network traffic. However, cores may be put into sleep mode and the power used by node v is equal to

$$P_v = P_v^{\text{UNIT}} \times \#\text{cores}$$

with P_v^{UNIT} being the energy consumption of a single core.

C. Layered Graph.

Following an idea similar to [16], we use a layered graph G^{L} that is defined as follows. We add $\max_{c \in C} n_c$ layers to the original graph G and each layer contains a copy of G. For every node $u \in V$, let v^i be the corresponding node in the ith layer ($i = 0, 1, \ldots, n_c$). Every $(i-1, i)$ layer pair is connected by (v^{i-1}, v^i) links. Provisioning of a chain and node placement of its functions amounts to find a path from node v_s on the first layer, i.e. v_s^0, to node v_d on the n_cth layer, i.e., $v_d^{n_c}$. Placement of a function on a node is given by the endpoints of the link used to switch between layers.

IV. Compact formulation

We now present the ILP formulation for the EE-SFCP problem. Let us first introduce the set of variables.

- $x_\ell \in \{0, 1\}$ where $x_\ell = 1$ if link ℓ is active, 0 otherwise
- $f_{i\ell}^{sd,c} \in \{0, 1\}$ where $f_{i\ell}^{sd,c} = 1$ if the flow for the request (v_s, v_d, c, D_{sd}^c) uses the link ℓ in layer i. We consider here un-splittable routing.
- $a_{iv}^{sd,c} \in \{0, 1\}$ where $a_{iv}^{sd,c} = 1$ if the ith function of the chain c is executed on node v for the request (v_s, v_d, c, D_{sd}^c).
- $k_v \in \mathbb{N}$, number of CPU cores used in node v.
- $f_\ell \in \mathbb{R}$, flow passing through link (u, v). This variable is linked and is added to the ILP for clarity of the presentation.

The formulation is given as follows.

Objective

$$\min \sum_{\ell \in L} \left(P_\ell^{\text{ON}} \times x_\ell + P_\ell^{\text{MAX}} \times \frac{f_\ell}{C_\ell^{\text{LINK}}} \right) + \sum_{u \in V} P_v k_u \quad (1)$$

Flow Conservation

$$\sum_{\ell \in \omega^+(v)} f_{i\ell}^{sd,c} - \sum_{\ell \in \omega^-(v)} f_{i\ell}^{sd,c} + a_{iv}^{sd,c} - a_{i-1v}^{sd,c} = 0 \quad (v_s, v_d) \in \mathcal{SD}, c \in C_{sd}, u \in V, 0 < i < n_c \quad (2)$$

$$\sum_{\ell \in \omega^+(v)} f_{0\ell}^{sd,c} - \sum_{\ell \in \omega^-(v)} f_{0\ell}^{sd,c} + a_{0v}^{sd,c} = \begin{cases} 1 \text{ if } u = v_s, \\ 0 \text{ else} \end{cases} \quad (v_s, v_d) \in \mathcal{SD}, c \in C_{sd}, u \in V \quad (3)$$

$$\sum_{\ell \in \omega^+(v)} f_{n_c\ell}^{sd,c} - \sum_{\ell \in \omega^-(v)} f_{n_c\ell}^{sd,c} - a_{n_c-1v}^{sd,c} = \begin{cases} -1 \text{ if } u = v_d, \\ 0 \quad \text{else} \end{cases} \quad (v_s, v_d) \in \mathcal{SD}, c \in C_{sd}, u \in V. \quad (4)$$

Link Capacity

$$f_\ell = \sum_{(v_s, v_d) \in \mathcal{SD}} \sum_{c \in C_{sd}} \sum_{i=0}^{n_c} D_{sd}^c \times f_{i\ell}^{sd,c} \leq C_\ell^{\text{LINK}} \times x_{uv} \quad \ell \in A. \quad (5)$$

Number of CPU cores used

$$\sum_{(s,t)\in\mathcal{D}} \sum_{i=0}^{n_c-1} \left(\Delta_{f_i^c} D_{sd}^c\right) \times a_{iv}^{sd,c} \leq k_v \qquad u \in V. \quad (6)$$

Node Capacity

$$k_v \leq C_v^{\text{NODE}} \qquad u \in V. \quad (7)$$

V. Solving large Instances with GreenChains

As the ILP proposed in the previous section cannot provide solutions for large networks, we propose here an ILP-based heuristic algorithm called GreenChains to solve the EE-SFCP problem. The problem can be decomposed into three sub-problems.

- First, the *energy saving problem* tries to put into sleep mode as many links and cores as possible to decrease the energy consumption of the network.
- Second, the *routing problem* computes a path for each request, respecting the link capacity constraints.
- Last, the goal of the *service chain placement problem* is to find a placement of the NVF respecting the capacities of the nodes and the order defined by the service chains, according to the path computed for each request.

A. Energy Saving Module.

The goal of this module is to put links into sleep mode.

It first launches the routing module and then places the network functions on the requests' paths. If both modules succeed, it creates a list U of all links according to their usage (volume of traffic). It then chooses the less loaded link $\ell_{\min}$ as a candidate to be put in sleep mode. It now considers the graph $G' = (V, L \setminus \{\ell_{\min}\})$. It launches the routing and placement modules again. If they succeed, $\ell_{\min}$ is put in sleep mode. The list U is actualized with the new routing, as well as the less loaded link. If at least one of the two modules fails, GreenChains considers that $\ell_{\min}$ cannot be into sleep mode and the link is kept active for the final solution. The second element of U is then considered. The algorithm goes on till all links have been tried and set either as definitely in sleep mode or active. The goal of this module is to reduce the energy used by the links by putting in sleep mode as many links as possible.

B. Routing Module

We consider the requests one by one and compute a weighted shortest path on a residual graph for each one of them. To favor links with a lower load, the weight of the link in the residual graph is equal to the inverse of its residual capacity. When we assign a path to a request, we decrease the capacity of the residual graph by the amount of charge requested. Furthermore, when considering a new demand to be routed, we remove links with a residual capacity smaller than the demand.

C. Service Chain Placement Module.

This module is in charge of choosing the execution location of the chains functions. We propose the following ILP that aims at minimizing the total number of cores used.

Given a path $P_{sd,c}$ for every request (v_s, v_d, c, D_{sd}^c), we need to find the execution location of each function of the chain c. Each node of the path is indexed by i, i.e., $P_{sd,c}^i$ is the ith node of $P_{sd,c}$.

We introduce the following two sets of variables.

- $a_{iv}^{sd,c} \in \{0,1\}$ where $a_{i,v}^{sd,c} = 1$ if f_i^c for request (v_s, v_d, c, D_{sd}^c) is executed on node v
- $k_v \in \mathbb{N}$, #cores used in node v.

The formulation is as follows.

Objective function

$$\min \sum_{u\in V} k_u. \quad (8)$$

Execution constraints

$$\sum_{v\in P_{sd,c}} a_{iv}^{sd,c} = 1 \quad (v_s, v_d) \in \mathcal{SD}, c \in C_{sd}, 1 \leq i \leq n_c. \quad (9)$$

Order constraints

$$a_{i,P_{sd,c}^k}^{sd,c} \leq \sum_{j=1}^{k} a_{i-1,P_{sd,c}^j}^{sd,c} \qquad (v_s, v_d) \in \mathcal{SD}, c \in C_{sd}, \\ 1 \leq k \leq \text{LEN}(P_{sd,c}), 1 \leq i \leq n_c. \quad (10)$$

Number of cores used

$$\sum_{(v_s,v_d)\in\mathcal{SD}} \sum_{c\in C} \sum_{i=1}^{n_c} \left(\Delta_{f_i^c} D_{sd}^c\right) \times a_{iv}^{sd,c} \leq k_v \qquad u \in V. \quad (11)$$

Node Capacity constraints

$$k_u \leq C_u \quad u \in V. \quad (12)$$

VI. Decomposition Models

As the ILP does not scale, we propose a column generation scheme to help validate our heuristic for larger networks. We first present here a model using Column Generation, *CG-simple*. We then introduce two variants of the models, *CG-cuts*, and *CG-cut+*. Indeed, problems dealing with energy-efficiency frequently lead to large integrality gap and bad precision. This is due to the On-Off phenomena of power models, which translates into large steps of the objective function. We thus try to improve the precision of the model by introducing different sets of constraints. We discuss the precision of the models in Section VII-C.

A. Column Generation Formulation

We propose a column generation formulation that relies on the concept of *Service Path*: each *Service Path* p is associated with a 4-uplet (v_s, v_d, c, D_{sd}^c) and defines: *(i)* a potential route for the request (v_s, v_d, c, D_{sd}^c) between v_s and v_d, *(ii)* node placement of the functions of chain c along the potential route. A route is described by parameters δ_ℓ^p, equal to the number of occurrences of the link ℓ in the path p. Node placement is

given by a^p_{vi}, equal to 1 if the ith function of the chain c is located at node v, 0 otherwise. We denote by P^c_{sd} the overall set of *Service Path* for each request (v_s, v_d, c, D^c_{sd}).

We now define the set of variables. First set of decision variables: $x_\ell = 1$ if link ℓ is on (active), 0 otherwise. Note that links are powered off by pair, i.e., $x_{\ell=(v,v')} = x_{\ell'=(v',v)}$. Second set of decision variables: $y^p_d = 1$ if demand d is routed using configuration p, 0 otherwise. Integer variables: $k_v = \#$ required cores in node v.

The objective, i.e., the minimization of the energy, can be written

$$\min \quad \underbrace{\sum_{\ell \in L} P^{\text{ON}}_\ell x_\ell}_{\substack{\text{link switch} \\ \text{on energy}}} + \underbrace{\sum_{\ell \in L} \sum_{p \in P^c_{sd}} \delta^p_\ell \left(\sum_{d=(v_s,v_d,c) \in D} \frac{D^c_{sd}}{C^{\text{LINK}}_\ell} P^{\max}_\ell \right) y^p_d}_{\text{link bandwidth energy}} + \underbrace{\sum_{u \in V} P_v \, k_v}_{\text{node resource energy}} \tag{13}$$

The constraint set decomposes into three sets of constraints.

One path per demand

$$\sum_{p \in P^c_{sd}} y^p_d = 1 \qquad (u_s, u_d) \in \mathcal{SD}, c \in C_{sd} \in D. \tag{14}$$

Link capacity

$$\sum_{d=(v_s,v_d,c) \in D} \sum_{p \in P^c_{sd}} D^c_{sd} \, \delta^p_\ell \, y^p_d \le x_\ell \, C^{\text{LINK}}_\ell \qquad \ell \in L. \tag{15}$$

Node capacity

$$\sum_{d \in D} \sum_{p \in P^c_{sd}} D^c_{sd} (\sum_{i=1}^{n_c} \Delta_{f_i} a^p_{v f_i}) y^p_d \le k_v \le C^{\text{NODE}}_v \qquad u \in V. \tag{16}$$

As we faced issues with large integrality gaps, we enhanced model (13)-(16) with different sets of cuts, through the next two models.

***CG-cuts* model.** The first set of cuts in (17) states that, for each node, at least one incident link should always be on. Moreover, the second inequality given by Equation (18) enforces that at least $n-1$ links should be active to have a connected network (or different if not all-to-all).

$$\sum_{\ell \in \omega^+(v)} x_\ell \ge 1 \qquad u \in V \tag{17}$$

$$\sum_{\ell \in L} x_\ell \ge n - 1. \tag{18}$$

***CG-cut+* model.** We further enhance the *CG-cuts* model with:

$$x_\ell \ge \sum_{p \in P^c_{sd}} \gamma^p_\ell \, y^p_d \qquad \ell \in L, (u_s, u_d) \in \mathcal{SD}, c \in C_{sd} \tag{19}$$

where $\gamma^p_\ell = 1$ if the link ℓ belong the path p. Using (14), it follows that $\sum_{p \in P^c_{sd}} \gamma^p_\ell \, y^p_d \le 1$. It avoids the use of a big M formulation at the expense of a large number of constraints.

B. Solution Scheme

To solve the model of Section VI-A efficiently, we need to recourse to column generation for solving the linear relaxation, and then to derive an ILP value, using the last restricted master problem. For additional details on linear programming and column generation schemes see, e.g., [17].

There is a configuration generator, i.e., pricing problem, for each request (v_s, v_d, c, D^c_{sd}). Two sets of decision variables are required. First set is made of variables φ^i_ℓ such that $\varphi^i_\ell = 1$ if the provisioning of demand d uses link ℓ in layer i of the layered graph G^{L}, 0 otherwise. Second set contains variables a^i_v such that $a^i_v = 1$ if the ith function (f^c_i) of chain c for request (v_s, v_d, c, D^c_{sd}) is placed on NFV node v, 0 otherwise. The formulation of the *Service Path* generator is given as follows.

$$\min -u^{(14)}_{sd} + \sum_{\ell \in L} \sum_{i=0}^{n_c} \varphi^i_\ell \times \left(P^{\max}_\ell \frac{D^c_{sd}}{C^{\text{LINK}}{}_\ell} + u^{(15)}_\ell D^c_{sd} \right) + \sum_{u \in V} \sum_{i=0}^{n_c - 1} a^i_v \times \left(u^{(16)}_v \Delta_{f_i} D^c_{sd} \right). \tag{20}$$

Path computation (flow conservation constraints):

$$\sum_{\ell \in \omega^+(v)} \varphi^i_\ell - \sum_{\ell \in \omega^-(v)} \varphi^i_\ell + a^i_v - a^{i-1}_v = 0 \qquad u \in V, 0 < i < n^c \tag{21}$$

$$\sum_{\ell \in \omega^+(v)} \varphi^0_\ell - \sum_{\ell \in \omega^-(v)} \varphi^0_\ell + a^0_v = \begin{cases} 1 \text{ if } v = v_s \\ 0 \text{ else} \end{cases} \qquad u \in V \tag{22}$$

$$\sum_{\ell \in \omega^+(v)} \varphi^{n_c}_\ell - \sum_{\ell \in \omega^-(v)} \varphi^{n_c}_\ell - a^{n_c}_v = \begin{cases} -1 \text{ if } v = v_d \\ 0 \text{ else} \end{cases} \qquad u \in V. \tag{23}$$

$$\text{Link capacity:} \quad D^c_{sd} \sum_{i=0}^{n_c} \varphi^i_\ell \le C^{\text{LINK}}_\ell \qquad \ell \in L. \tag{24}$$

$$\text{Node capacity:} \quad D^c_{sd} \sum_{i=0}^{n_c} \Delta_{f_i} a^i_v \le C^{\text{NODE}}_v \qquad u \in V. \tag{25}$$

1) Speeding up the Pricing Problem: The Pricing Problem corresponds to a constrained shortest path with negative

weights on the layered graph, and we can use CPLEX to solve it. However, if we discard the capacity constraints, the problem becomes the *shortest path with negative weights* problem. It can be solved much faster than the original problem using the Bellman-Ford shortest path algorithm. Since we remove the capacity constraints, the set of solutions considered is a superset of the initial set of solutions. It is possible to find a path that might use more resources than available. In this case, we fall back the ILP solver to obtain a valid path. The weight of the inter-layer arcs is thus given by

$$w_{iv} = P_\ell^{\max} \frac{D_{sd}^c}{C^{\text{LINK}}{}_\ell} + u_\ell^{(15)} D_{sd}^c \qquad 0 \leq i < n_c, u \in V.$$

and the weight of intra-layer arcs by

$$w_{i\ell} = u_v^{(16)} \Delta_{f_i} D_{sd}^c \qquad 0 \leq i \leq n_c, \ell \in L.$$

2) Particularities of CG-cut+*:* By introducing the constraints (19) into the model, we also need to introduce a new set of variable γ_l into the Pricing Problem that indicates if the link ℓ is used in the path. The objective function becomes

$$\begin{aligned} \min \quad -u_{sd}^{(14)} &+ \sum_{\ell \in L} \sum_{i=0}^{n_c} \varphi_\ell^i \times \left(P_\ell^{\max} \frac{D_{sd}^c}{C^{\text{LINK}}{}_\ell} + u_\ell^{(15)} D_{sd}^c \right) \\ &+ \sum_{u \in V} \sum_{i=0}^{n_c} a_v^i \times \left(u_v^{(16)} \Delta_{f_i} D_{sd}^c \right) \\ &+ \sum_{\ell \in L} \gamma_l u_{sd,c,l}^{(19)}. \end{aligned} \tag{26}$$

and the link capacities constraints become

$$D_{sd}^c \sum_{i=0}^{n_c} \varphi_\ell^i \leq C_\ell^{\text{LINK}} \times \gamma_\ell \qquad \ell \in L. \tag{27}$$

Moreover, adding enhanced cuts creates negative cycles in the layered graph used for the Pricing Problem. We choose not to get rid of the cycles by enumerating them all. Instead, we check if the solution provided by the solver contains any negative cycles. If that is the case, we add the corresponding constraints in the formulation and call the solver again. We repeat this process until the obtained solution no longer contains any negative cycles or the reduced cost is no longer negative. Removing the cycle has a negligible impact on the performance of the column generation scheme as it is executed only a few times at the start of the algorithm.

VII. Numerical Experiments

In this section, we investigate the energy savings obtained by the Column Generation model. We compare the results with the ones of GreenChains heuristic algorithm. We first present the data sets we use for the experiments. We then take a look at the precision of the solutions obtained by the Column Generation model and GreenChains. We investigate different improvements of the model presented in Section III. We then present the energy savings achieved for network topologies of different sizes. Last, we discuss the impact of the solutions on link usage and path lengths.

A. Data sets

In networks, each type of flows has to go through a different chain of network services. In our experiments, we consider four of the most frequent types of flows, as presented in Table I: Video Streaming, Web Service, Voice-over-IP (VoIP), and Online Gaming. The traffic percentages are from [18]. For each one, we give the ordered set of functions required and the bandwidth used. In total, we have six different functions, and each function requires a different amount of cores to be executed.

We tested the CG models and GreenChains on three topologies of different sizes from SNDlib [19]: *pdh* (11 nodes and 64 directed links), *atlanta* (15 nodes and 44 directed links), and *germany50* (50 nodes and 176 directed links).

To obtain realistic looking traffic matrices, we generate, for each network, a set of demands from the traffic matrices provided in SNDlib: we divide each aggregate flow from a source to a destination into four demands corresponding to the four different types of traffic. We consider that the flows provided in the SNDlib data set represent aggregated flows of miscellaneous services. Thus, we can subdivided them into four different services. The original load of the flow is conserved, and each sub-flow load is given by the distribution of the last column of Table I. For example, a flow with a charge of 1 is split into a Web Service, a VoIP, a Video Streaming and an Online Gaming sub-flows with a load of 0.182, 0.118, 0.699 and 0.001, respectively.

We tested the solution on a daily traffic to see how much energy can be saved during the day or at night. The variations of traffic come from a trace of a typical France Telecom link shown in Figure 2. Previous work [20] indicates that using a small number of configurations during the day is enough to obtain most of the energy savings. In our case, we considered five different levels of traffic called D1 to D5. D1 represents the period with the lowest amount of traffic and D5 the one with the highest.

Traditionally, ISP networks use shortest path routing and operate their network with an overprovisioning factor of 2 or 3 [21], [22], in order to be able to cope with failures and traffic growth. This means that links typically are used at only between 30 and 50% of their capacity. We set capacities accordingly at the beginning of the simulation. For each network, we solve the legacy scenario by routing requests on the shortest paths between each location of the service's functions. Each function location is chosen at random in the legacy scenario. We then choose the link capacities such that each link is at most used at 33% of its capacity. Finally, we considered equal values for the energy consumption of the links and nodes.

B. Compact formulation evaluation

We compare the results obtained by the heuristic algorithm, GreenChains, with the optimal results given by the integer linear program on a small network, *pdh*, with 11 nodes and 64 links. We consider instances with an increasing complexity: the number of demands varies from 4 to 40. Note that we consider multiples of 4 demands, as the traffic between a pair

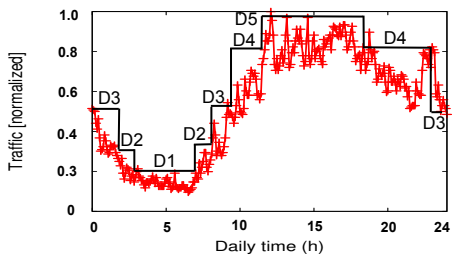


Fig. 2: Normalized daily variation of traffic of a France Telecom network link and multi-period approximation

Service Chains	Chained VNFs	Rate	traffic (%)
Web Service	NAT-FW-TM-WOC-IDPS	100 kbps	18.2
VoIP	NAT-FW-TM-FW-NAT	64 kbps	11.8
Video Streaming	NAT-FW-TM-VOC-IDPS	4 Mbps	69.9
Online Gaming	NAT-FW-VOC-WOC-IDPS	50 kbps	0.1

TABLE I: Service Chain Requirements [3]

(a) Execution time

(b) Energy used

Fig. 3: Comparison between the compact formulation and GREENCHAINS

of nodes is divided into four different demands corresponds to different categories of traffic.

We compare the execution times of the ILP model and the algorithm in Figure 3a. The experiments are made on a Intel Xeon E5620 with 24GB of RAM. We see that the ILP model can be used to solve the problem with a reasonable amount of time for a maximum number of 16 demands. In this case, it takes around 45 minutes to return the optimal solution. The increase is exponential: for 20 demands, the execution time is almost 3 hours. On the other hand, GREENCHAINS is much faster as it can find a solution in less than 1 second for 20 demands (0.38 s). It solves an instance with 40 demands in 0.78 s and the all-to-all instance (with 440 demands), considered in the following, in less than 7 s. We see that the ILP cannot be used in practice to solve instances with a large number of demands, and thus we use the GREENCHAINS for the experiments on larger networks in the following.

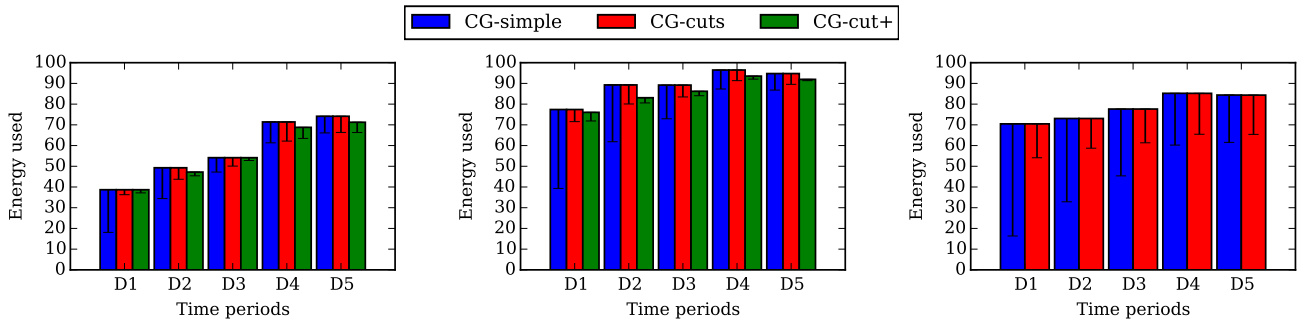
The results regarding energy savings are given in Figure 3. GREENCHAINS finds results within a precision ranging between 0% and 16% for the different number of demands. We consider this as good results given the difficulty of the EE-SFCP problem. Moreover, it means that the potential energy savings of using dynamic traffic and virtualization are in fact even greater than the one presented in the following.

C. Quality of the Column Generation models

We now compare the performance of the three different CG models (*CG-simple*, *CG-cuts*, and *CG-cut+*) with respect to their accuracy as given by $\varepsilon = (\tilde{z}_{\text{ILP}} - z^{\star}_{\text{LP}})/z^{\star}_{\text{LP}}$, where $z^{\star}_{\text{LP}}$ represents the optimal value of the relaxation of the Restricted Master Problem, and $\tilde{z}_{\text{ILP}}$ the integer solution obtained at the end of the column generation algorithm. In Figure 4, 5, and 6, we compare the solutions found by the three CG models for all three networks and for the 5 different levels of traffic. We first observe in Figure 4, in which error bars represent the gap between the relaxed and integer solutions, that *CG-simple* and *CG-cuts* provide similar solutions. However, ε dramatically varies, as shown in Figure 5. Cuts significantly improves ε: for *CG-simple*, it varies between 12% and 113% for pdh, 10% and 97% for atlanta, and 37% and 330% for germany50. For *CG-cuts*, ε is between 7 to 15% for pdh, 6 and 12% for atlanta, and 24 and 30% for germany50. The ratio is further improved with *CG-cut+*: between 4 and 8% for pdh, 1 and 6% for atlanta. However, no solutions were found in a reasonable amount of time for the germany50 topology. As the energy savings are similar for the three models, it shows that the *three CG models provide rather accurate solutions, as confirmed by the solutions and accuracy of the* CG-cuts *and* CG-cut+ *models.*

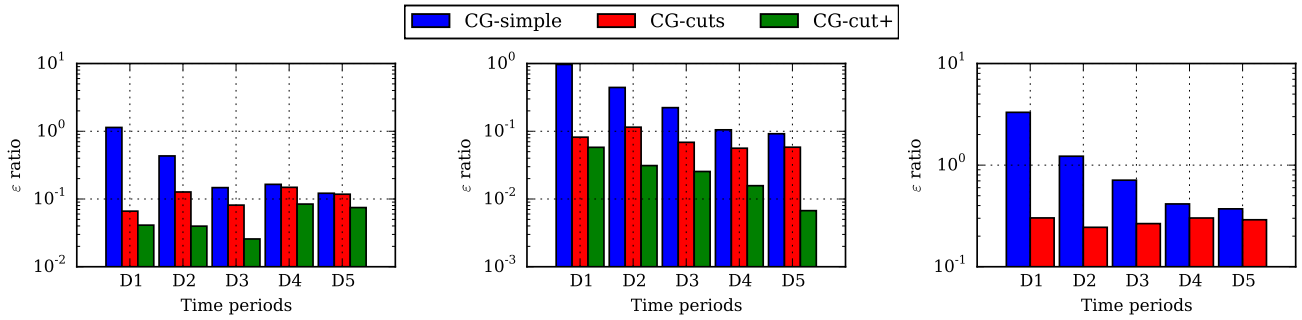
Finally, in Figure 6, we compare the execution times of the models. We observe that *CG-cut+* execution time (between 17 s and 5 h) is orders of magnitude higher that the one of *CG-simple* (between 50 ms and 440 s) and *CG-cuts* (between 70 ms and 670 s). This is greatly due to the fact that we speed up the resolution of the two previous model using the Bellman-Ford shortest path algorithm for the Pricing Problem. The second factor is that *CG-cut+*'s cuts slow the convergence time of the column generation drastically.

We now *focus on the* CG-cuts *model*, as it offers the best compromise in terms of accuracy (w.r.t. *CG-simple* model) and computation time requirements (w.r.t. *CG-cut+* model) to solve large networks.



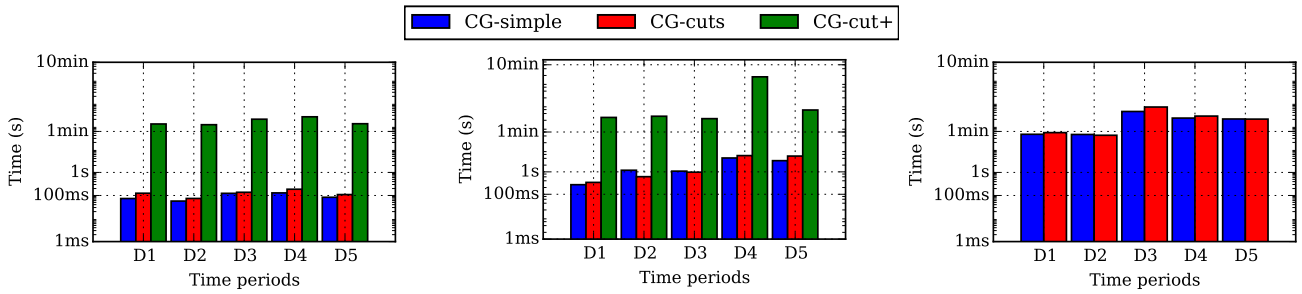
(a) pdh (b) atlanta (c) germany50

Fig. 4: Performance of all three CG models on the (a) *pdh*, (b) *atlanta* and, (c) *germany50* network topologies. 100 represents the energy used by the legacy scenario.



(a) pdh (b) atlanta (c) germany50

Fig. 5: Accuracy, ε, of all three CG models on the (a) *pdh*, (b) *atlanta* and, (c) *germany50* network topologies



(a) pdh (b) atlanta (c) germany50

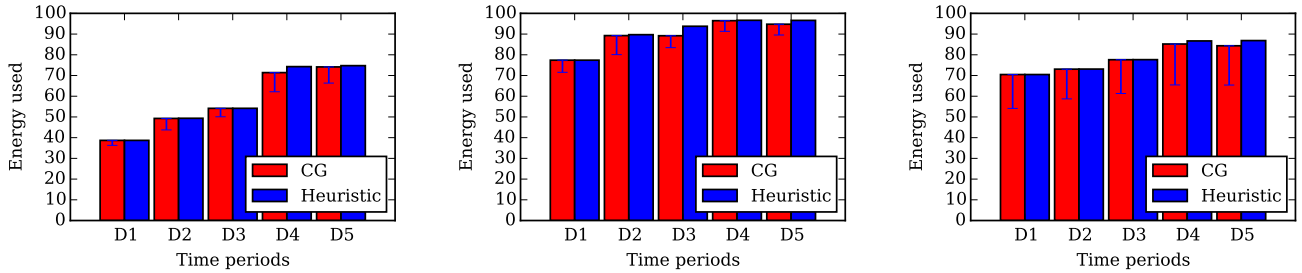
Fig. 6: Execution times of all three CG models on the (a) *pdh*, (b) *atlanta* and, (c) *germany50* network topologies

D. Energy Savings

We now compare the energy savings obtained by GREEN-CHAINS and *CG-cuts*. We consider three scenarios in the experiments:

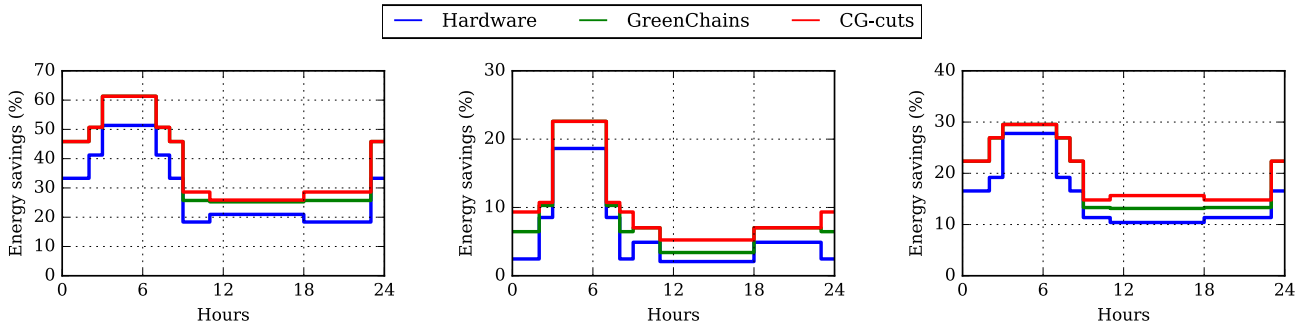
- *Legacy scenario.* This scenario corresponds to the one of a legacy network, whose operator does not try to reduce the energy consumption of its network. Its goal is to minimize the total bandwidth used while respecting the link capacity and the chain constraints. This scenario is used as a *baseline for comparison* for the energy-aware algorithms.
- *Hardware scenario.* The hardware scenario corresponds to one of an SDN (non-virtualized) network in which an operator tries to reduce its energy consumption by adapting the routing to the demands. In this scenario, the network functions are carried out by some specific hardware placed at given positions in the network.
- *NFV scenario.* The NFV scenario is the one of a virtualized SDN network in which generic hardware nodes can execute any virtual network functions. This is the scenario solved by the solutions provided in Sections IV, V and VI.

We provide in Figure 7 the energy used for the five levels of demands for *pdh*, *atlanta*, and *germany50*. The values are normalized: 100 corresponds to the legacy scenario. We also present in Figure 8 the corresponding energy savings during the day. We see that we obtained important savings using virtualization: between 25 and 61% for *pdh*, 5 and 22% for *atlanta*, and 15 and 30% for *germany50*.



(a) pdh (b) atlanta (c) germany50

Fig. 7: Energy used for GREENCHAINS and the CG model on the three topologies.



(a) pdh (b) atlanta (c) germany50

Fig. 8: Saved energy for GREENCHAINS for the three network topologies.

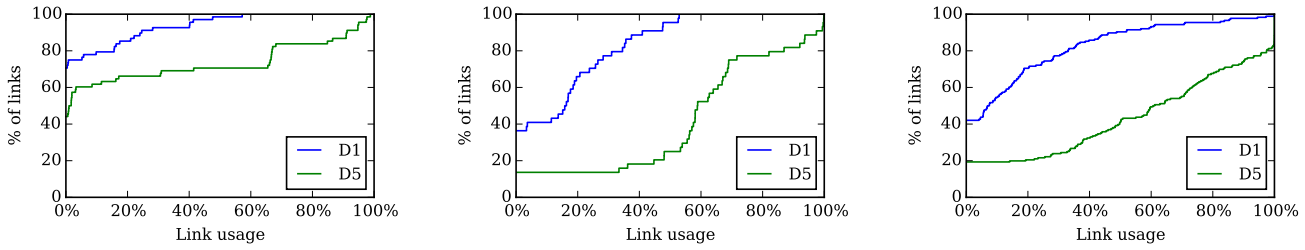
1) Validating GREENCHAINS *with* CG-cuts*:* We now compare the solutions provided by both GREENCHAINS and *CG-cuts* in Figure 7. Error bars on the *CG-cuts* solutions represent the lower bounds given by $z^{\star}_{\text{LP}}$. For the lowest traffic periods (D1, D2 and in D3), both methods provide similar solutions for *pdh* and *germany50*. The CG model provides slightly better solutions when the traffic is higher, with a difference of 3 and 1% for *pdh*, of 5,and 2 for *atlanta*, and of 2 and 3% for *germany50* respectively in the D5 period. Observe that, even if CG only provides slight improvement of the heuristic's solutions, it shows (c.f. ε accuracy value) that the heuristic gives good results, regardless of the traffic period.

2) Link load: To reduce the amount of energy used by the network, we reroute some of the flows to be able to put links into sleep mode. This means that the remaining active links are more loaded. In Figure 9, we look at the link load given by GREENCHAINS for the highest and lowest traffic periods. First, we see that, unsurprisingly, the percentage of links with no traffic is higher when the traffic is low, around 40% of the links for *atlanta* and *germany50*. When the network is at its highest utilization, it drops to around 15% for both networks. The *pdh* network, due to its higher link density, can have more links put into sleep mode. Indeed, between 44% and 71% of the links have no traffic. Moreover, at the lowest traffic period, no links are used at 100% for *pdh*, *atlanta* and are at most used up to 57%, 52% of their capacity, respectively. At rush hour, *pdh* and *atlanta* have at most links at 98 and 99% capacity while *germany50* has only one link at full capacity.

3) Impact on Delay: When some links are put into sleep mode, paths tend to become longer. However, we show in Figure 10 that the maximum delay of every path stays below the usual 50 ms latency value in Service Level Agreements: experienced delay is less than 5.4, 10.8 and 16.2 ms on *pdh*, *atlanta* and *germany50* respectively. Moreover, the median of the delay stays constant for *pdh*, *atlanta* at 3.6 and 5.4 ms, respectively. For *germany50*, it only increases from 7.2 for D5 (no link into sleep mode) to 9 ms for D1.

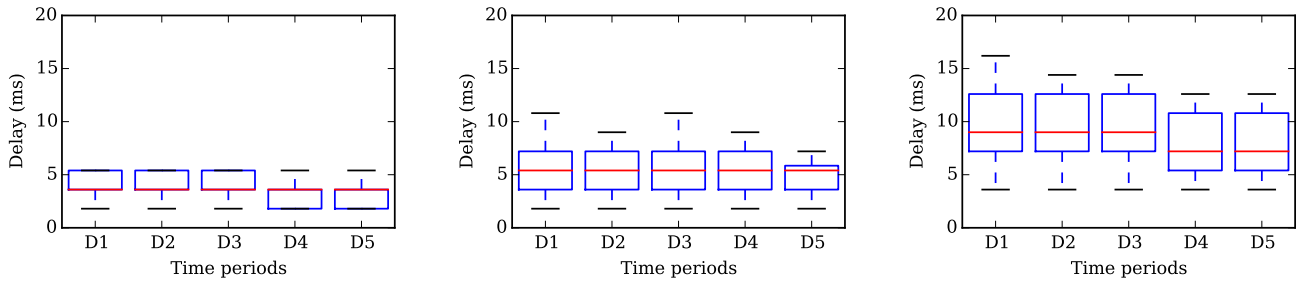
VIII. CONCLUSIONS

In this work, we investigate the potential of network virtualization to reduce the energy consumption of networks. We introduce a Column Generation model to solve the problem of minimizing network energy consumption while satisfying the SFC requirements. We also propose GREENCHAINS, an ILP-based heuristic that we validate using our Column Generation model. We then compare three different scenarios corresponding to a continuous deployment of the SDN and NFV paradigm for energy efficiency. We show that an operator, using SDN control, can save energy by choosing the paths of the flows dynamically according to the variations of demands during the day. Indeed, this allows the turning off of a large portion of network equipment. Indeed, compared to a legacy scenario, SDN can provide between 18 and 51% energy savings during the night. We also demonstrated that the deployment of VNF in an SDN network leads to additional energy savings between 4 and 12%. As a matter of fact, choosing dynamically the locations of network functions according to the variations of the demands allows a greater flexibility for the choice of the network paths and leads to the use of less network equipment.



(a) pdh (b) atlanta (c) germany50

Fig. 9: Link load for GREENCHAINS for the three network topologies.



(a) pdh (b) atlanta (c) germany50

Fig. 10: Delay in milliseconds for GREENCHAINS for the three network topologies.

REFERENCES

[1] D. Matsubara, T. Egawa, N. Nishinaga, V. P. Kafle, M.-K. Shin, and A. Galis, "Toward future networks: a viewpoint from itu-t," *IEEE Communications Magazine*, vol. 51, no. 3, pp. 112–118, 2013.

[2] W. Vereecken, W. Van Heddeghem, M. Deruyck, B. Puype, B. Lannoo, W. Joseph, D. Colle, L. Martens, and P. Demeester, "Power consumption in telecommunication networks: overview and reduction strategies," *IEEE Communications Magazine*, vol. 49, no. 6, 2011.

[3] M. Savi, M. Tornatore, and G. Verticale, "Impact of processing costs on service chain placement in network functions virtualization," in *IEEE Conference on Network Function Virtualization and Software Defined Network (NFV-SDN)*, Nov. 2015, pp. 191–197.

[4] A. Gupta, M. Habib, P. Chowdhury, M. Tornatore, and B. Mukherjee, "On service chaining using virtual network functions in network-enabled cloud systems," in *IEEE International Conference on Advanced Networks and Telecommuncations Systems (ANTS)*, 2015, pp. 1–3.

[5] A. Gupta, B. Mukherjee, B. Jaumard, and M. Tornatore, "Service chain (SC) mapping with multiple SC instances in a wide area network," in *IEEE Global Telecommunications Conference - GLOBECOM*, 2017, pp. 1–6.

[6] B. Martini, F. Paganelli, P. Cappanera, S. Turchi, and P. Castoldi, "Latency-aware composition of virtual functions in 5G," in *1st IEEE Conference on Network Softwarization (NetSoft)*, 2015, pp. 1–6.

[7] A. Mohammadkhan, S. Ghapani, G. Liu, W. Zhang, K. Ramakrishnan, and T. Wood, "Virtual function placement and traffic steering in flexible and dynamic software defined networks," in *21st IEEE Intl. Workshop on Local and Metropolitan Area Networks*, 2015, pp. 1–6.

[8] R. Riggio, A. Bradai, T. Rasheed, J. Schulz-Zander, S. Kuklinski, and T. Ahmed, "Virtual network functions orchestration in wireless networks," in *Intl. Conf. on Network and Service Management (CNSM)*, 2015, pp. 108–116.

[9] L. Chiaraviglio, M. Mellia, and F. Neri, "Minimizing ISP network energy cost: formulation and solutions," *IEEE/ACM Transactions on Networking (TON)*, vol. 20, pp. 463–476, April 2012.

[10] F. Giroire, J. Moulierac, and K. Phan, "Optimizing rule placement in software-defined networks for energy-aware routing," in *IEEE Global Telecommunications Conference - GLOBECOM*, Austin, USA, December 2014, pp. 2523–2529.

[11] F. Giroire, J. Moulierac, T. K. Phan, and F. Roudaut, "Minimization of network power consumption with redundancy elimination," *Computer communications*, vol. 59, pp. 98–105, 2015.

[12] R. Bolla, C. Lombardo, R. Bruschi, and S. Mangialardi, "DROPv2: energy efficiency through network function virtualization," *IEEE Network*, vol. 28, no. 2, pp. 26–32, 2014.

[13] R. Mijumbi, "On the energy efficiency prospects of network function virtualization," *CoRR*, vol. abs/1512.00215, 2015. [Online]. Available: http://arxiv.org/abs/1512.00215

[14] J. Chabarek, J. Sommers, P. Barford, C. Estan, D. Tsiang, and S. Wright, "Power awareness in network design and routing," in *IEEE Annual Joint Conference of the IEEE Computer and Communications Societies - INFOCOM*, April 2008, pp. 1130–1138.

[15] L. Niccolini, G. Iannaccone, S. Ratnasamy, J. Chandrashekar, and L. Rizzo, "Building a power-proportional software router," in *USENIX Annual Technical Conference (USENIX ATC)*, Boston, MA, USA, 2012, pp. 89–100.

[16] A. Dwaraki and T. Wolf, "Adaptive service-chain routing for virtual network functions in software-defined networks," in *Workshop on Hot topics in Middleboxes and Network Function Virtualization (HotMIddlebox)*, 2016, pp. 32–37.

[17] V. Chvatal, *Linear Programming*. Freeman, 1983.

[18] I. C. V. Networking, "Cisco visual networking index: Forecast and methodology 2015-2020," *White paper, CISCO)*, 2015.

[19] S. Orlowski, M. Pióro, A. Tomaszewski, and R. Wessäly, "SNDlib 1.0–Survivable Network Design Library," *Networks*, vol. 55, no. 3, pp. 276–286, 2010.

[20] J. Araujo, F. Giroire, J. Moulierac, Y. Liu, and R. Modrzejewski, "Energy efficient content distribution," *The Computer Journal*, vol. 59, no. 2, pp. 192–207, Feb. 2016.

[21] C. Fraleigh, S. Moon, B. Lyles, C. Cotton, M. Khan, D. Moll, R. Rockell, T. Seely, and S. Diot, "Packet-level traffic measurements from the sprint IP backbone," *IEEE network*, vol. 17, no. 6, pp. 6–16, 2003.

[22] S. Iyer, S. Bhattacharyya, N. Taft, and C. Diot, "An approach to alleviate link overload as observed on an IP backbone," in *IEEE Annual Joint Conference of the IEEE Computer and Communications Societies - INFOCOM*, vol. 1, 2003, pp. 406–416.